

Creating a Multi-tier Architecture for Web Applications: Design and Implementation

Danylo Sereda^{*}

B.Sc. in Automation and Computer-Integrated Technologies, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

Email: danilseredapersonal@gmail.com

Abstract

In today's world, where web applications are becoming more complex and diverse, the need for a layered architecture is particularly relevant. This architecture offers flexibility, scalability and ease of management, which is extremely important in the context of rapidly increasing requirements for information systems. Designing and implementing effective architectural solutions is becoming a crucial factor for the successful development and operation of web applications. Developers often face the limitations of a monolithic architecture, where every change in the code can unpredictably affect the entire system. This leads to difficulties in project management, difficulties in adding new features, and problems scaling the application. As a result, the risk of system failures increases and the rate of implementation of new features decreases. The study shows how a layered architecture can significantly reduce the dependency between system components, improve testability, and simplify the implementation of changes.

Keywords: multilevel architecture; web applications; design; implementation; system.

1. Introduction

Modern web applications play a critical role in various areas of life, providing users with access to complex systems and services at their fingertips. However, rising expectations for functionality and performance make developing and maintaining such applications increasingly challenging. One of the key solutions in designing sustainable and efficient systems is the use of a multi-tier architecture that offers a structured approach to managing complexity and change.

Received: 4/4/2025

Accepted: 5/28/2025

Published: 6/7/2025

* Corresponding author.

This paper aims to demonstrate how the design and implementation of a multi-tier architecture can significantly improve the structure of web applications, increasing their efficiency and making them more resilient to change. To achieve this goal, we set several objectives: first, to explore the basic principles and benefits of applying a multi-tier architecture; second, to elaborate on the steps involved in designing and implementing such an architecture, from conceptual schemes to the specific technologies and tools used at each tier.

In addition, we will pay special attention to analyzing examples of successful implementation of multi-tier architecture in real web applications. This will allow us to illustrate theoretical provisions and show the practical results of this approach. Finally, the article offers an assessment of the achieved improvements in performance and manageability, which will be an important criterion for organizations considering a move to a multi-tier architecture.

Thus, the article aims not only to provide recommendations for software developers and architects but also to increase understanding of the potential benefits and challenges of implementing multi-tier structures in modern Web application development.

2. Principles of layered architecture

The layered architecture paradigm represents one of the key concepts in software development. In this context, an optimal architecture is essential to simplify and improve the efficiency of software product development and maintenance processes. This architectural approach facilitates the software structure's extension, modification, testing, debugging, and understanding [1]. As a consequence, we can distinguish several universal criteria characterizing the architecture quality (fig.1.).

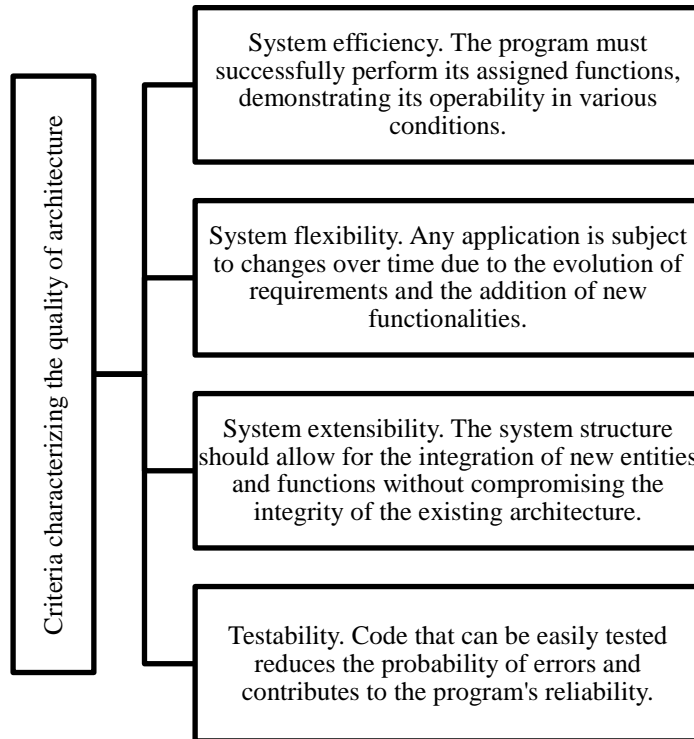


Figure 1: Criteria characterizing the quality of architecture [3]

These criteria are fundamental in assessing the quality of architectural solutions. They can be a basis for choosing the most appropriate architectural paradigm for developing complex software systems.

In software development, there is a concept of architectural design known as layered architecture. Its essence lies in the functional division of the application into independent components [3].

This approach has several distinctive features. First, each functional area strictly corresponds to its purpose. Second, the functional areas of the software are clearly delineated, from the user interface to the data access layer, including the service layer and business logic. Third, the functionality implemented on one level is not repeated on others, which eliminates code redundancy.

The interaction between the architecture levels is carried out through well-defined channels of information exchange while maintaining a low degree of connectivity. Each higher level combines and abstracts the functionality of the layer directly subordinate to it, avoiding mixing code from other functional areas.

The database component includes CRUD functions (create, read, update, delete) that manipulate information in the repository [2]. The architectural solution provides for flexible deployment on a single computer and in a distributed environment on multiple computers.

One key advantage of such a structure is the possibility of its integration and reuse in various software systems, which significantly increases the efficiency of development [4].

The functionality of the application is implemented at the level of business logic. According to the requirements of the technical specification, an ergonomic interface for user interaction is provided by the user interface layer.

3. Stages of designing and implementing a multi-level architecture

Designing and implementing a multi-level architecture involves several critical stages that ensure both scalability and maintainability of complex systems. Figure 2 below shows the stages of the development and subsequent implementation of a layered architecture for web applications.

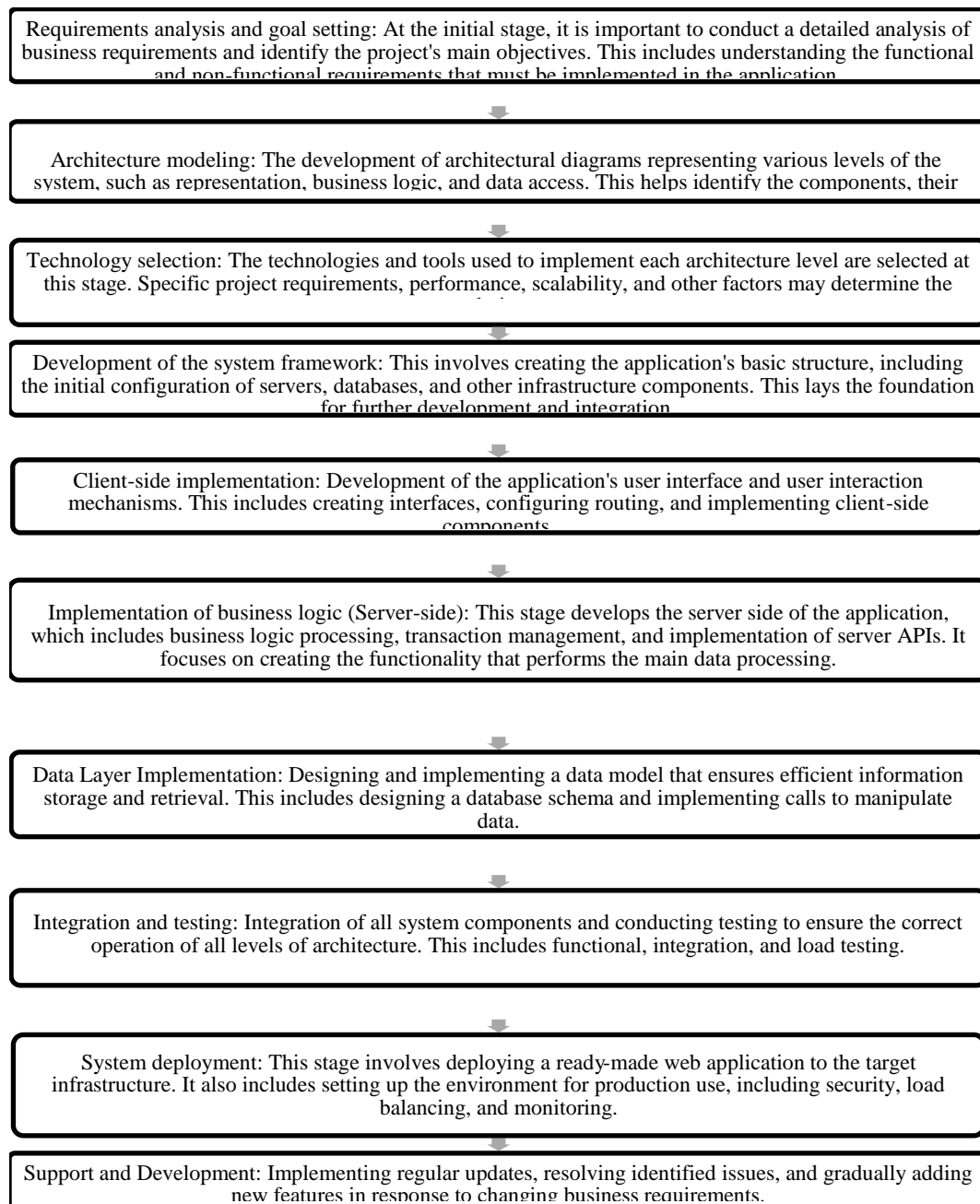


Figure 2: Stages of the development process and subsequent implementation of a multi-level architecture for web applications [6].

Following these steps will help ensure the creation of a web application with a layered architecture that meets the requirements of modern developments while providing flexibility, scalability, and resilience to change.

4. Introduction of modern technologies into the management system

Let us look at some successful examples of the use of complex architecture in the field of web applications:

Netflix successfully serves millions of users and offers customized recommendations based on their viewing history. This is achieved through a scalable and fault-tolerant architecture, where each microservice handles a specific aspect of the application's operation.

Airbnb uses a serverless architecture to regulate its complex booking and payment system. Each component functions autonomously in this model and can scale automatically to handle increasing workloads efficiently.

Uber manages its car-sharing platform using a complex system consisting of microservices and an event-driven architecture. This allows the company to quickly adapt routes and trip costs while processing millions of orders daily.

Spotify is implementing a scalable and adaptive architecture to manage its music streaming platform. Individual microservices handle specific functions within the application, which allows Spotify to provide services to millions of users and create customized music offerings based on their previous preferences.

GitHub uses a complex multi-level management system for its platform, which hosts code and collaborates with developers. This solution allows users to effectively manage millions of repositories and support the interaction of programmers worldwide [5, 7, 8].

5. Conclusion

This article provides an overview of the layered architecture, emphasizing the importance of structuring the application into key layers: user interface, business logic, application server, and database. Each of these layers plays its own unique role in ensuring the software's functionality and performance, while their interaction critically affects the efficiency of the entire system.

We also reviewed the main approaches to designing and implementing multi-level applications, including decomposition, which allows users to divide complex tasks into more straightforward and manageable components. Dividing responsibilities between levels not only simplifies the development process but also increases the level of support and scalability of the application in the future. The use of interoperability techniques such as APIs and other mechanisms ensures that changes in one layer do not affect the rest, which is a key aspect of ensuring the flexibility of the architecture.

To summarize, a multi-level architecture is a powerful tool in software development that allows for reliability, scalability, and convenience in application support. Implementing the presented approaches and practices in the

development process contributes to the creation of better and more efficient software that meets modern requirements and user expectations.

References

- [1]. Jesse Russell Fabregas (2023) *Cesc*. - M.: VSD. 372 p.
- [2]. V. Budilov (2020) *Fundamentals of programming for the Internet*. - Moscow: BHV-Petersburg. 736 p.
- [3]. T.S. Mashnin (2024) *Eclipse. Development undecipherable -, undecipherable -, undecipherable - append to undecipherable*. - M. BHV-Petersburg. 384 p.
- [4]. J. Forcier (2024). *Django. Web application development in Python*. M.: Symbol-plus. 364 p.
- [5]. L. Shklyar (2021). *Architecture of Web applications*. M.: Eksmo. 640 p.
- [6]. D. Esposito (2023). *Programming using Microsoft ASP.NET 4. Developing web applications using ASP.NET and AJAX (set of 2 books)*. M.: Petersburg. 529 p.
- [7]. V. V. Nikulin (2023). *Development of the server side of a web resource. A textbook for universities*. M.: Lan. 132 p.
- [8]. A.V. Dikov (2020). *Client-side web programming technologies: JavaScript and DOM*. M.: Lan. 124 p.